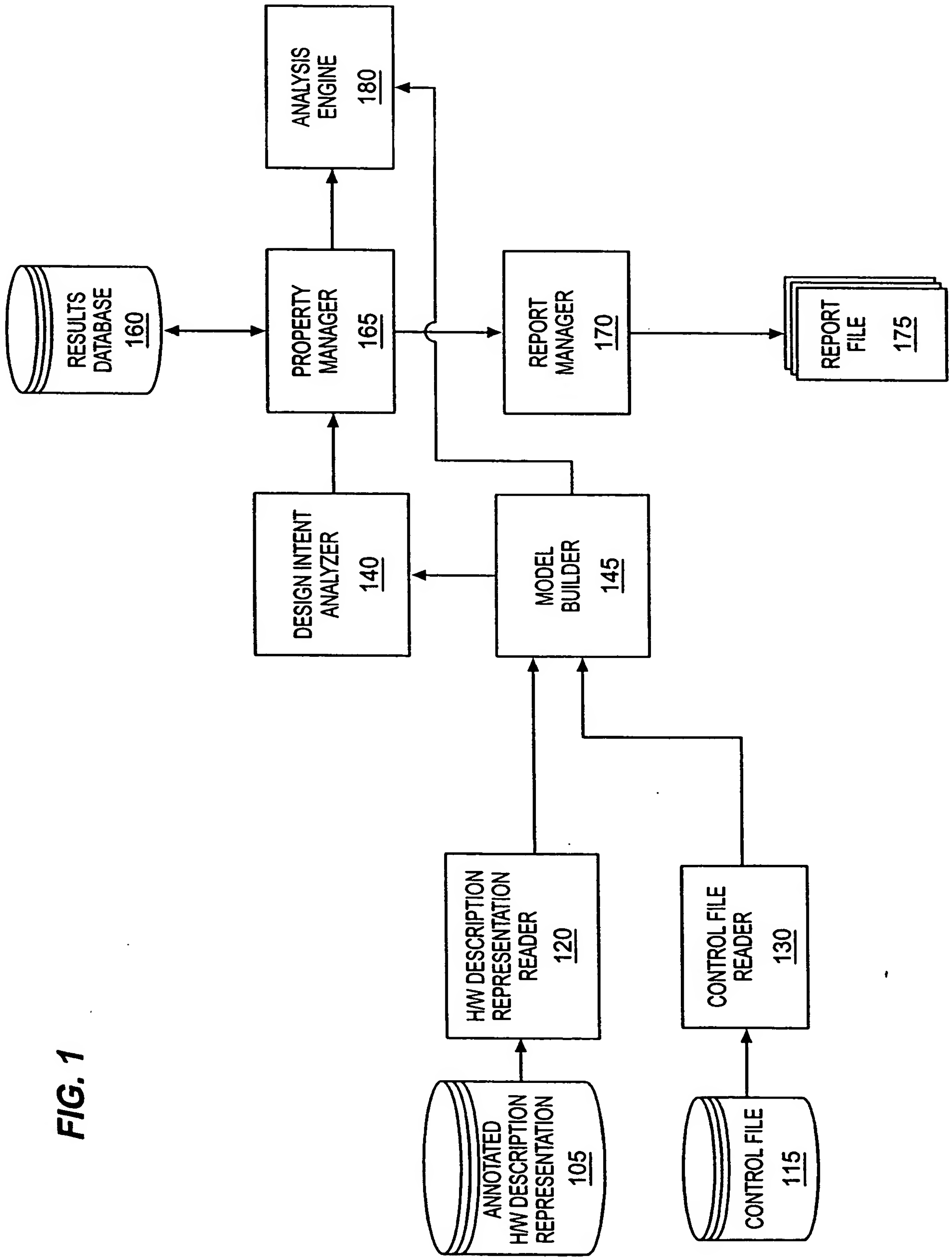
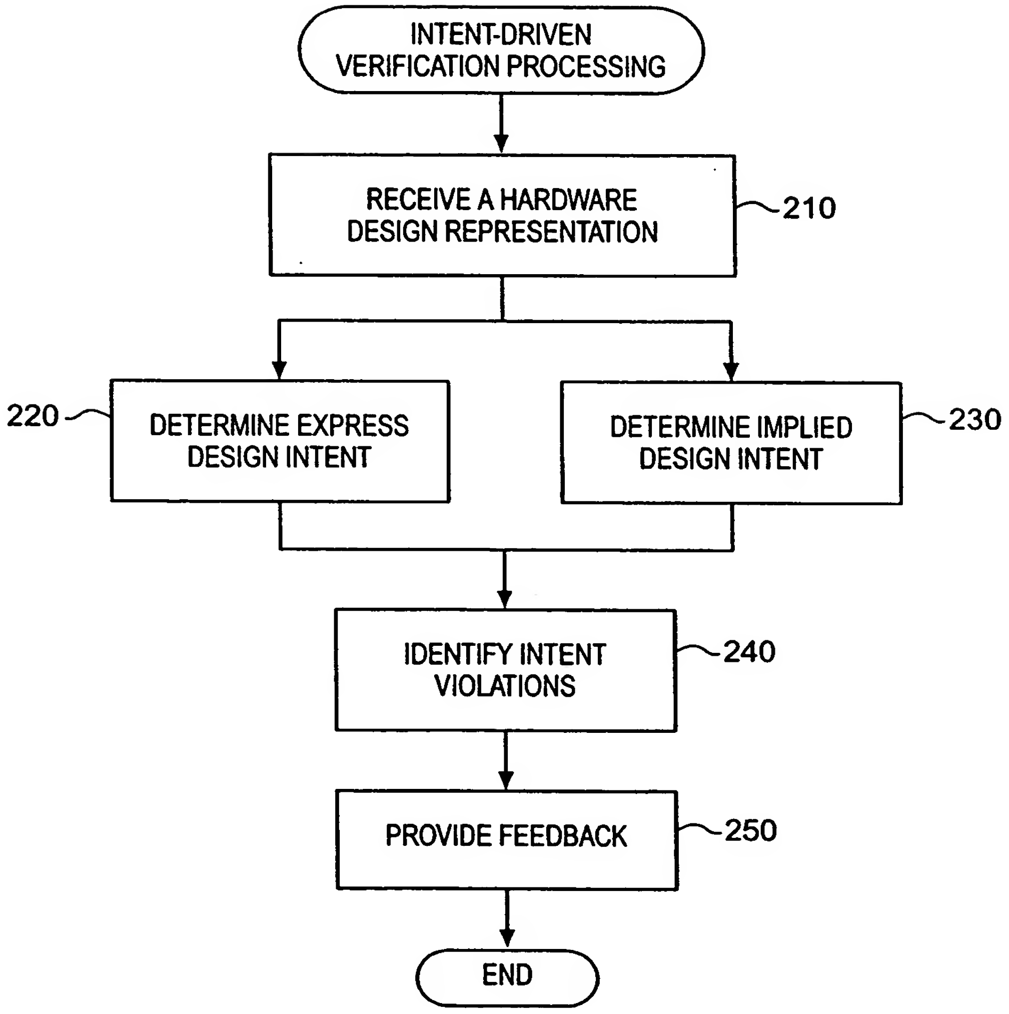

ANALYSIS ENGINE 180
RESULTS DATABASE 160
PROPERTY MANAGER 165
REPORT MANAGER 170
REPORT FILE 175
DESIGN INTENT ANALYZER 140
MODEL BUILDER 145
H/W DESCRIPTION REPRESENTATION READER 120
CONTROL FILE READER 130
ANNOTATED H/W DESCRIPTION REPRESENTATION 105
CONTROL FILE 115

FIG. 1

FIG. 2


INTENT-DRIVEN VERIFICATION PROCESSING
RECEIVE A HARDWARE DESIGN REPRESENTATION
210
DETERMINE EXPRESS DESIGN INTENT
220
DETERMINE IMPLIED DESIGN INTENT
230
IDENTIFY INTENT VIOLATIONS
240
PROVIDE FEEDBACK
250
END

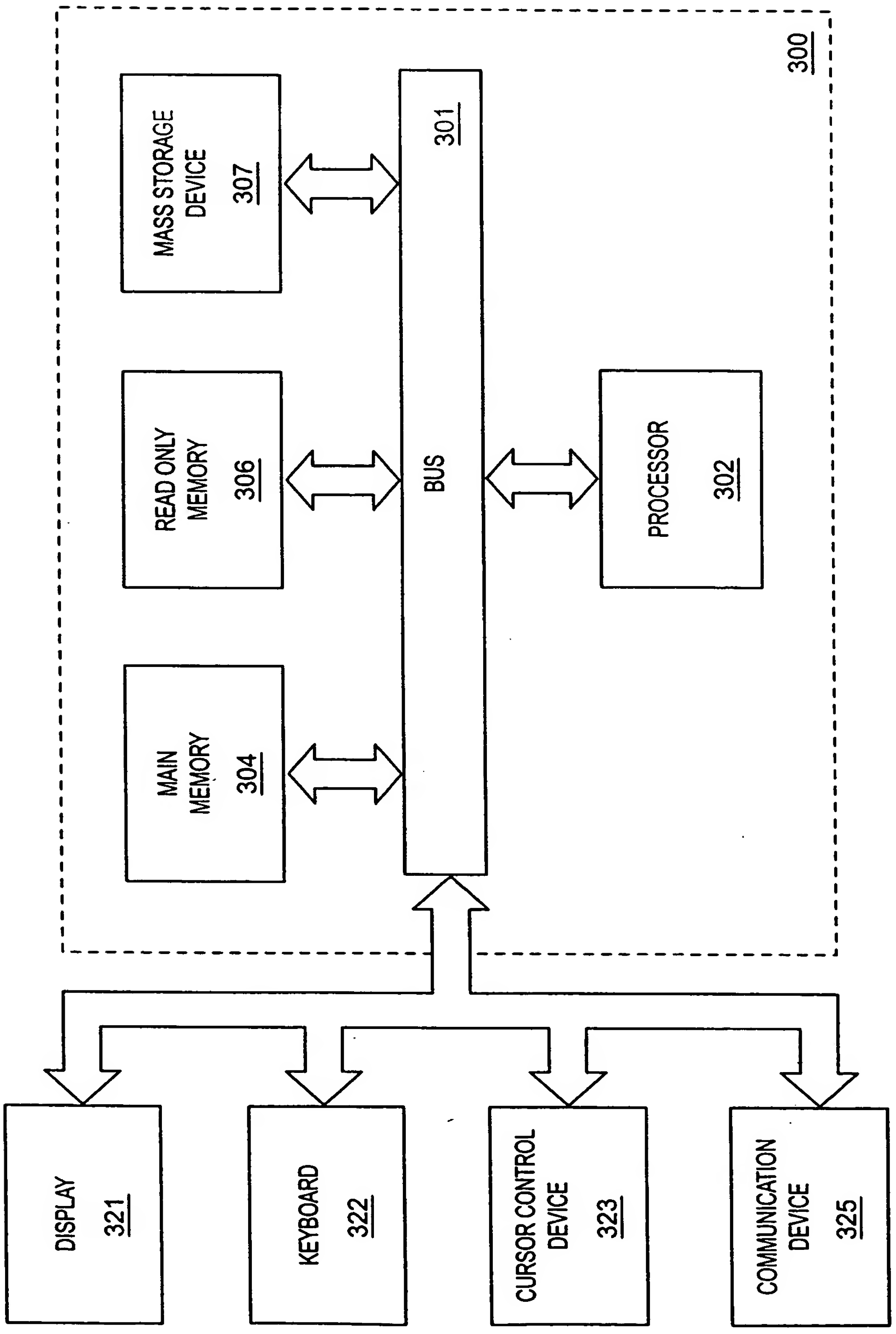

300
MAIN MEMORY 304
READ ONLY MEMORY 306
MASS STORAGE DEVICE 307
BUS 301
PROCESSOR 302
DISPLAY 321
KEYBOARD 322
CURSOR CONTROL DEVICE 323
COMMUNICATION DEVICE 325

FIG. 3

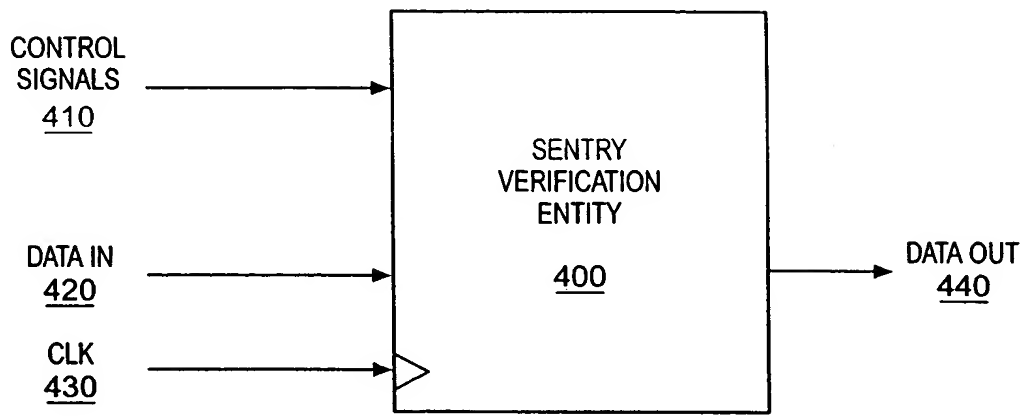

CONTROL SIGNALS 410
DATA IN 420
CLK 430
SENTRY VERIFICATION ENTITY 400
DATA OUT 440

FIG. 4A

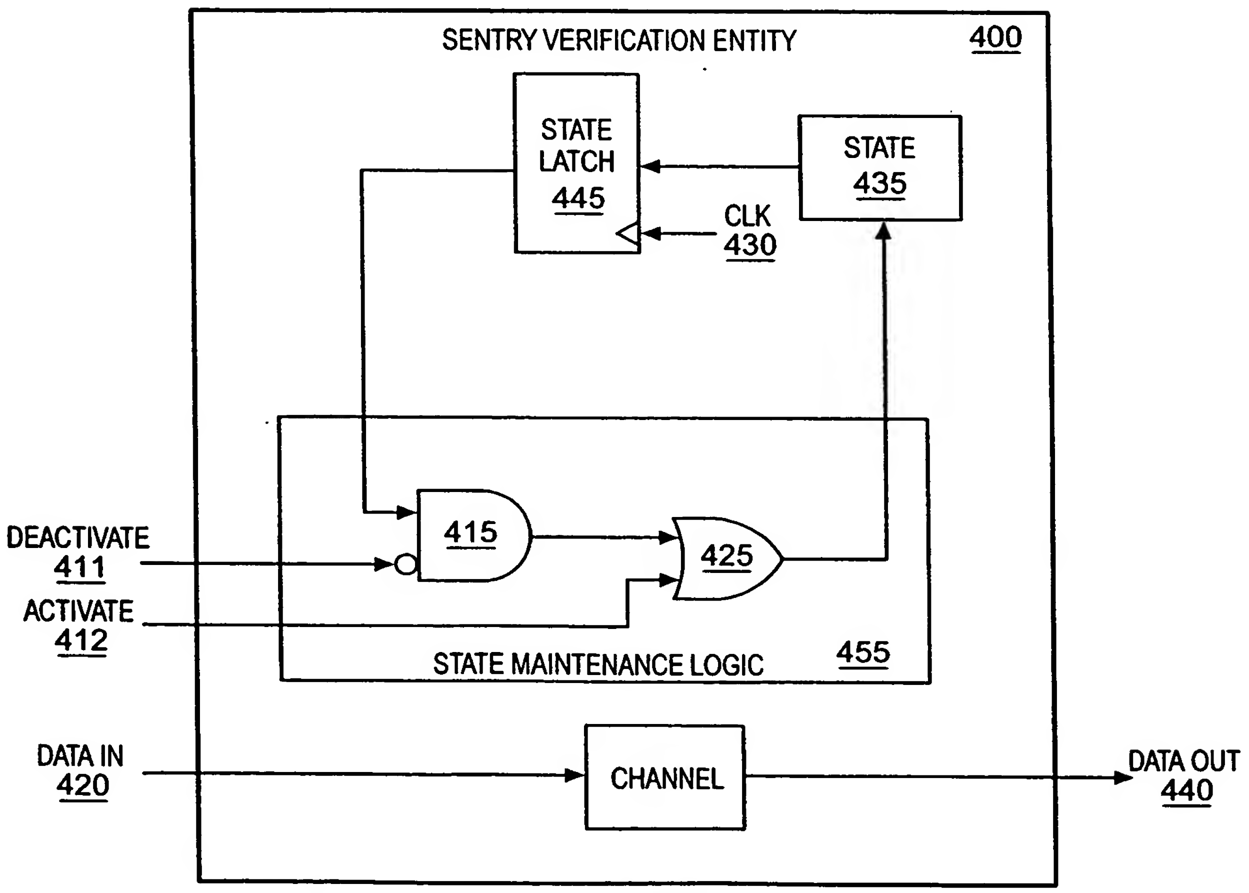

SENTRY VERIFICATION ENTITY 400
STATE LATCH 445
STATE 435
CLK 430
415
425
DEACTIVATE 411
ACTIVATE 412
STATE MAINTENANCE LOGIC 455
DATA IN 420
CHANNEL
DATA OUT 440

FIG. 4B

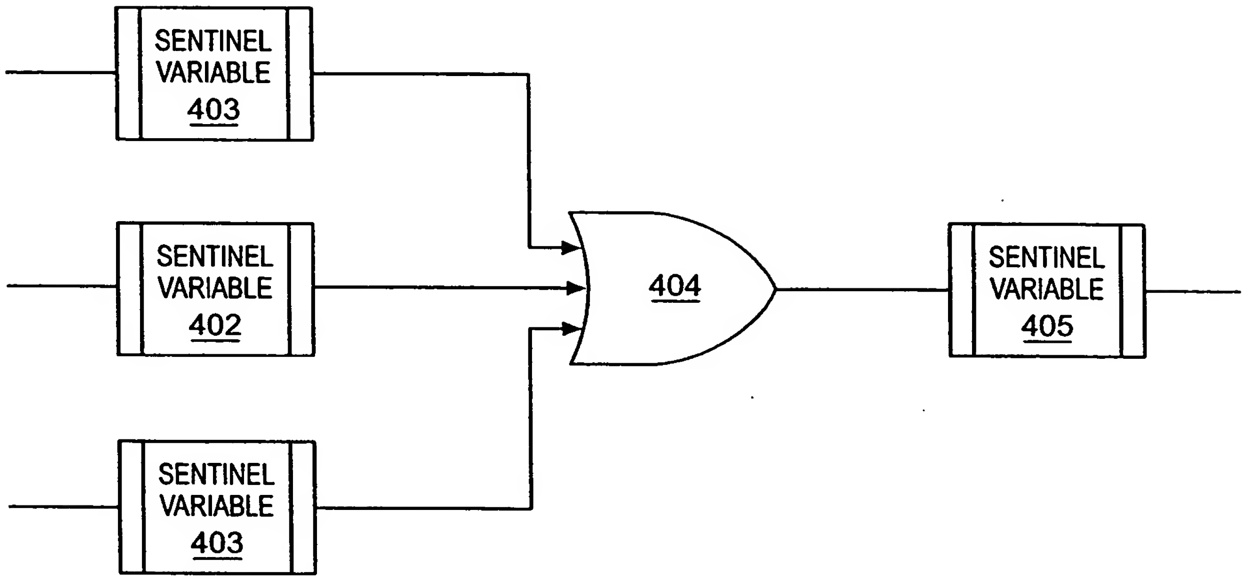
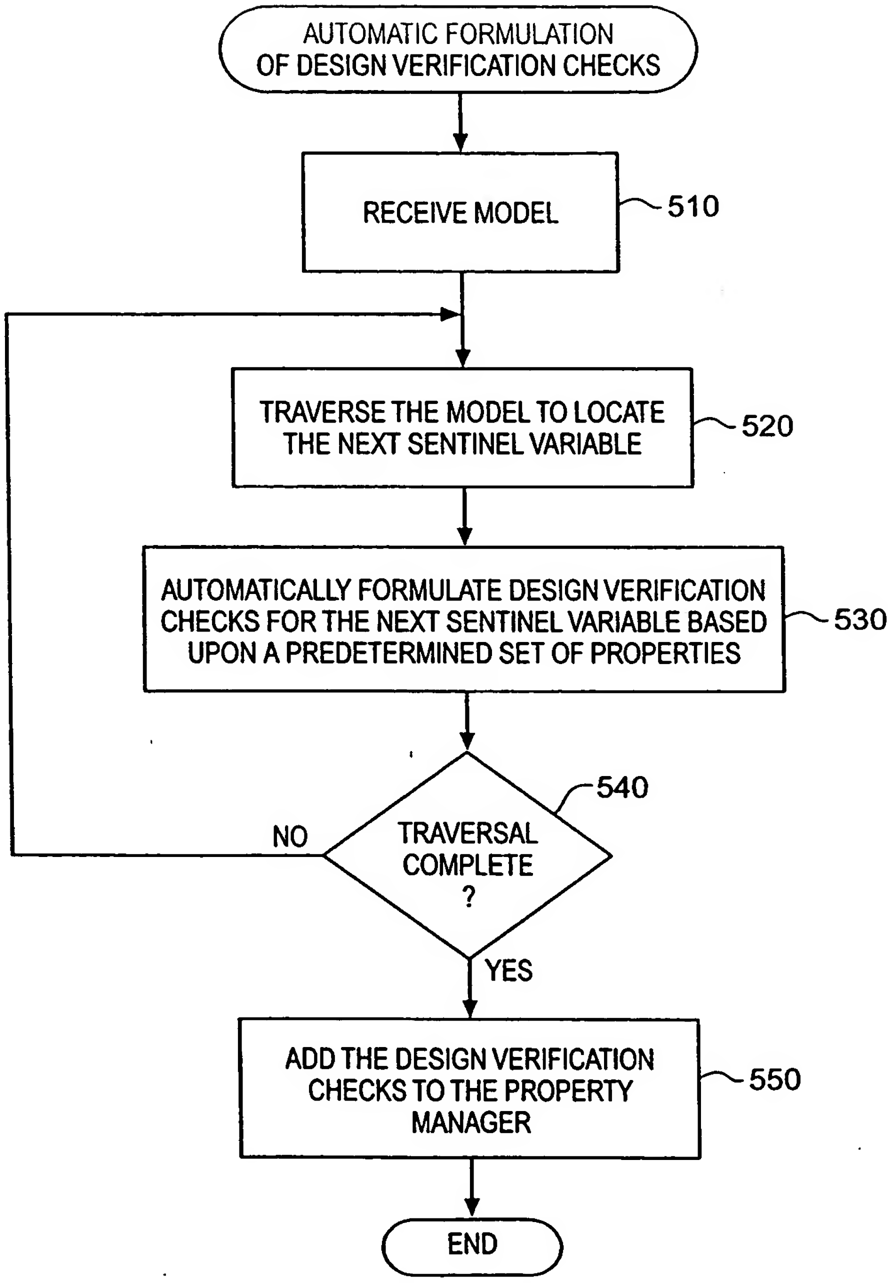

SENTINEL VARIABLE 403
SENTINEL VARIABLE 402
SENTINEL VARIABLE 403
404
SENTINEL VARIABLE 405

FIG. 4C

FIG. 5


AUTOMATIC FORMULATION OF DESIGN VERIFICATION CHECKS
RECEIVE MODEL
510
TRAVERSE THE MODEL TO LOCATE THE NEXT SENTINEL VARIABLE
520
AUTOMATICALLY FORMULATE DESIGN VERIFICATION CHECKS FOR THE NEXT SENTINEL VARIABLE BASED UPON A PREDETERMINED SET OF PROPERTIES
530
TRAVERSAL COMPLETE ?
540
NO
YES
ADD THE DESIGN VERIFICATION CHECKS TO THE PROPERTY MANAGER
550
END

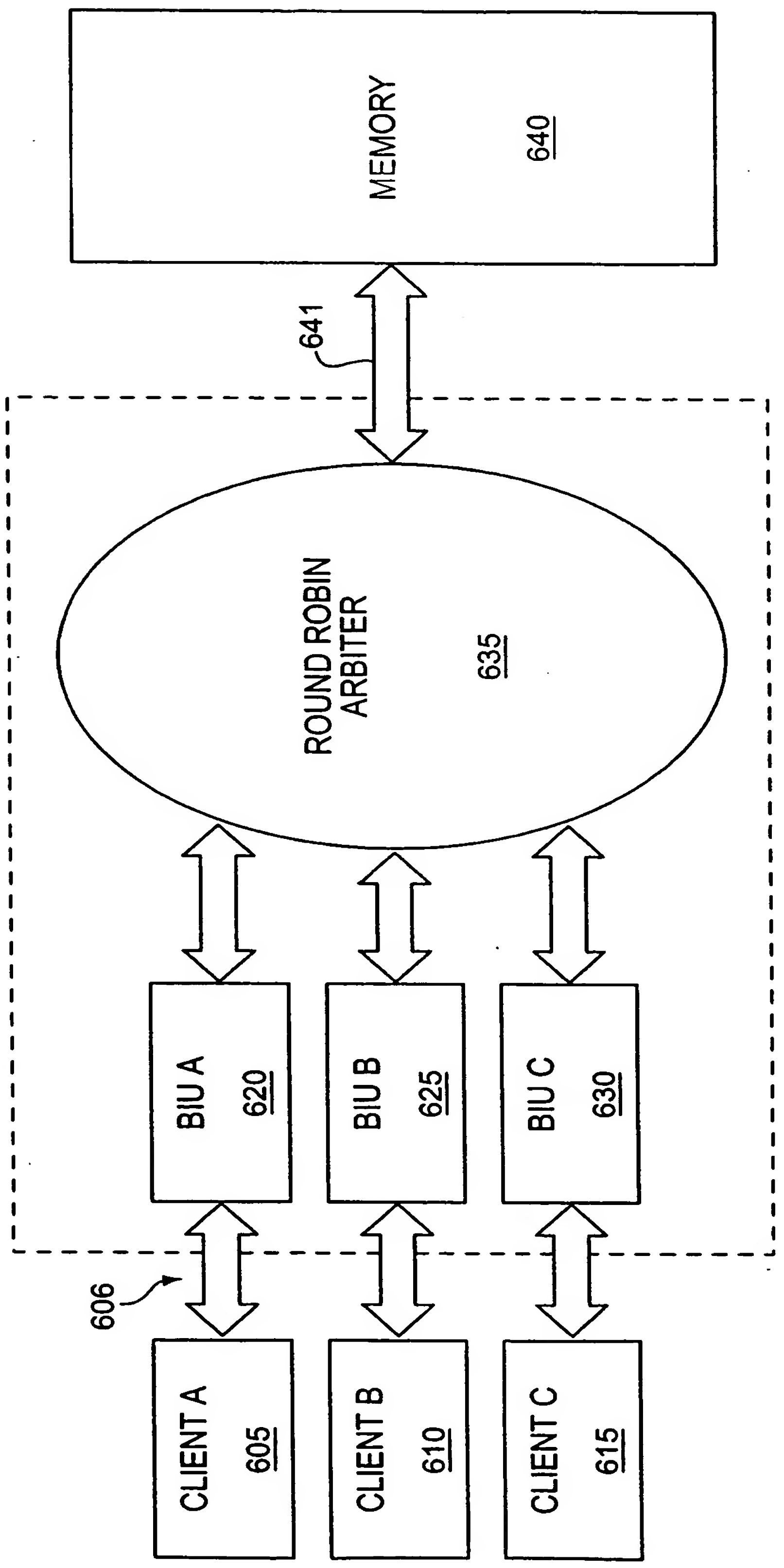

MEMORY
640
641
ROUND ROBIN ARBITER
635
BIU A
620
BIU B
625
BIU C
630
606
CLIENT A
605
CLIENT B
610
CLIENT C
615

FIG. 6A

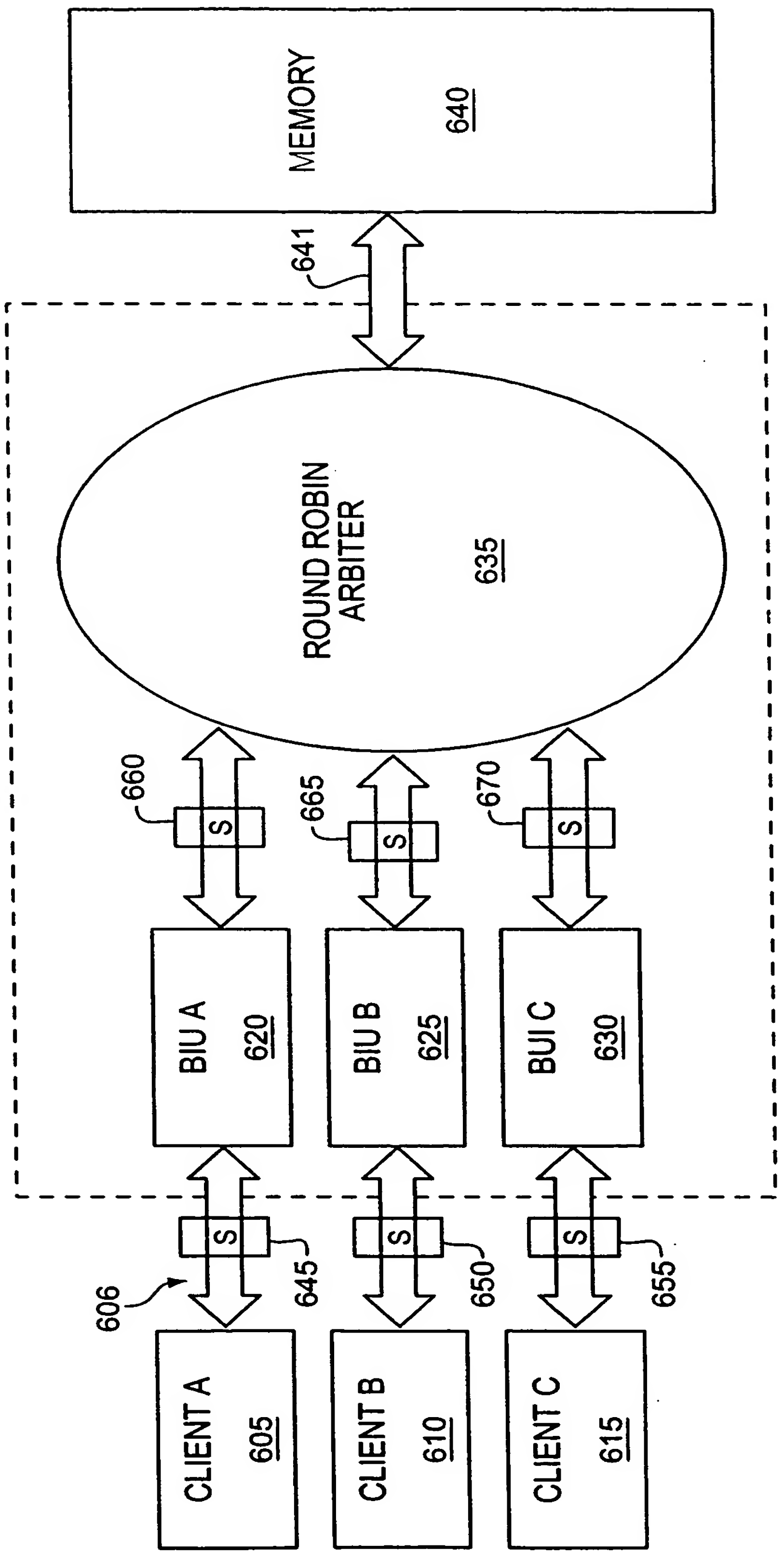

MEMORY
640
641
ROUND ROBIN ARBITER
635
660
S
665
S
670
S
BIU A
620
BIU B
625
BUI C
630
S
645
S
650
S
655
606
CLIENT A
605
CLIENT B
610
CLIENT C
615

FIG. 6B

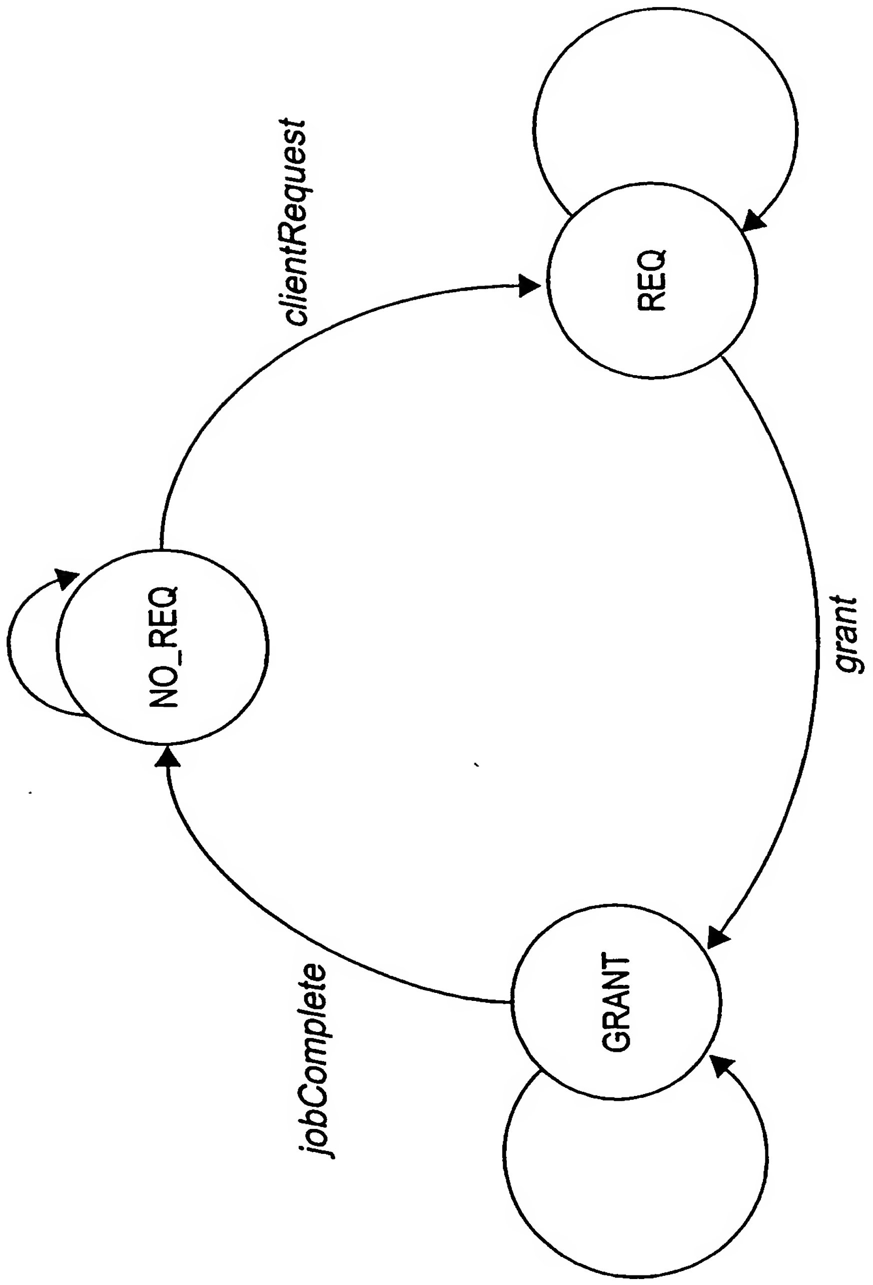


FIG. 7

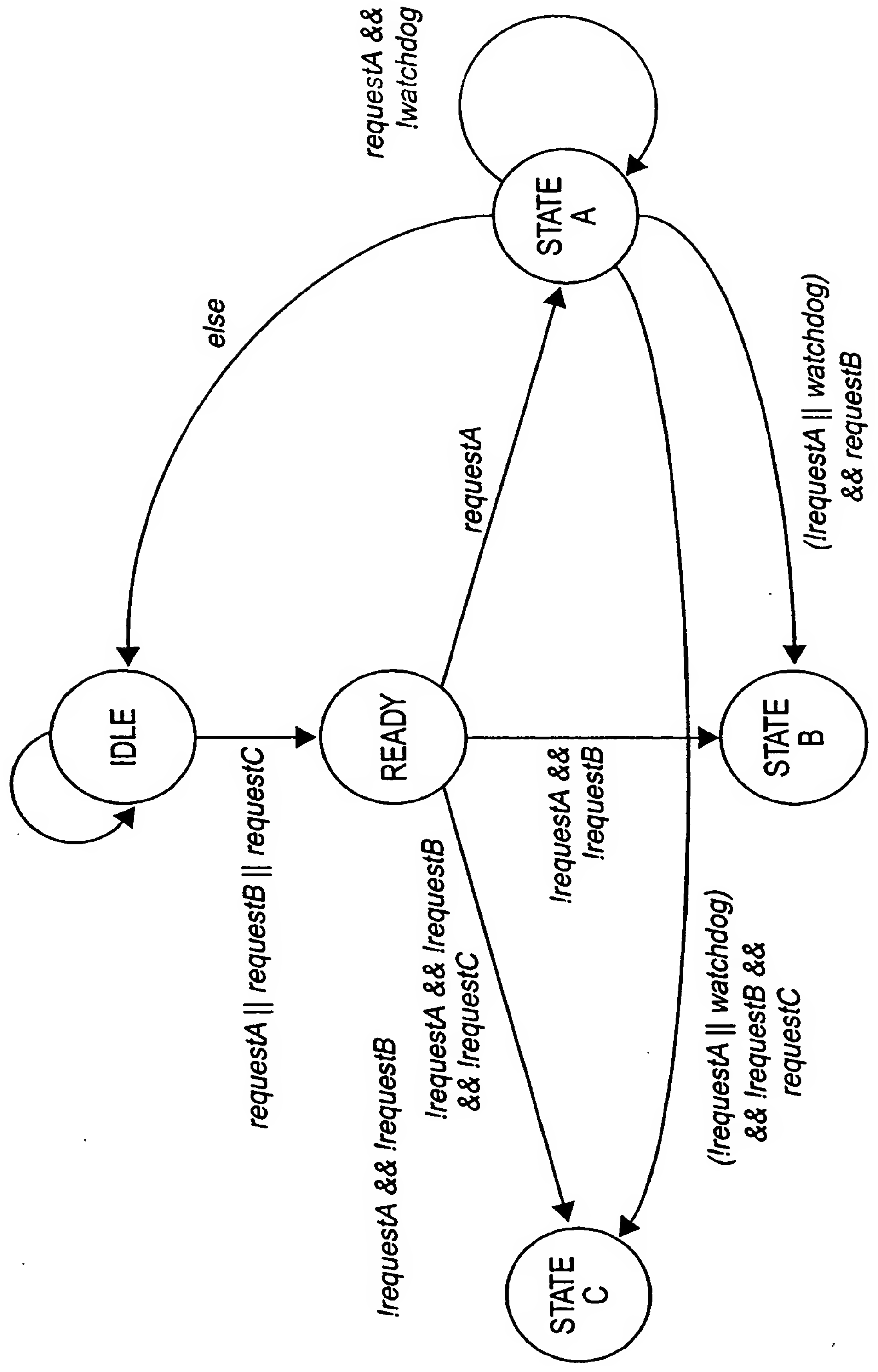

IDLE
READY
STATE A
STATE B
STATE C
requestA && !watchdog
else
requestA
(!requestA || watchdog) && requestB
requestA || requestB || requestC
!requestA && !requestB
!requestA && !requestB && !requestC
(!requestA || watchdog) && !requestB && requestC
!requestA && !requestB

FIG. 8

```
define DEFAULT_MAX_ACCESS_TIME 1'b1

module main(clk, reset,
            dataA, clientRequestA, jobCompleteA, clientGrantA,
            dataB, clientRequestB, jobCompleteB, clientGrantB,
            dataC, clientRequestC, jobCompleteC, clientGrantC,
            memoryData);

            input           clk, reset;
            input [0:0]     dataA, dataB, dataC;
            input           clientRequestA, clientRequestB, clientRequestC;
            input           jobCompleteA, jobCompleteB, jobCompleteC;
            output          clientGrantA, clientGrantB, clientGrantC;
            output [0:0]    memoryData;

            wire [0:0]      dataOutA, dataOutB, dataOutC;

            // Put sentries at the boundary signals and make them active constantly.

            // vx sentry dataA, dataB, dataC:clk;
            // vx always activate(dataA,dataB,dataC);

            // Create three instances of the bus interface

            BusInterface busInterfaceA(clk, reset, clientRequestA, JobCompleteA,
                  dataA, dataOutA, requestA, grantA,
                  clientGrantA);

            BusInterface busInterfaceB(clk, reset, clientRequestB, jobCompleteB,
                  dataB, dataOutB, requestB, grantB,
                  clientGrantB);

            BusInterface busInterfaceC(clk, reset, clientRequestC, JobCompleteC,
                  dataC, dataOutC, requestC, grantC,
                  clientGrantC);

            RoundRobinArbiter arbiter(clk, reset,
                  dataOutA, requestA, grantA,
                  dataOutB, requestB, grantB,
                  dataOutC, requestC, grantC,
                  memoryData);

endmodule // main
```

FIG. 9A

```
module BusInterface(clk, reset, clientRequest, jobComplete,
        data, dataOut, request, grant, clientGrant);

        input           clk, reset, clientRequest, jobComplete;
        input [0:0]     data;
        output [0:0]    dataOut;
        // vx sentry dataOut:clk;

        output          request;
        input           grant;
        output          clientGrant;

        parameter       NO_REQ = 2'b00;
        parameter       REQ=2'b01;
        parameter       GRANTED = 2'b10;

        reg [1:0] state, nextState;
        // vx flop state;

        //assign request = ((state == REQ) || (state == GRANTED));
        assign request = ((state == REQ) || ((state == GRANTED) && !jobComplete));
        assign dataOut = data;
        assign clientGrant = grant;

        always @(state or reset or clientRequest or jobComplete or grant)
        begin
                if (reset)
                begin
                        nextState = NO_REQ;
                        // vx deactivate(dataOut);
                end
                else
                begin
                        nextState = state;
                        case (state)
                                NO_REQ:
                                begin
                                        // vx deactivate(dataOut);
                                        if (clientRequest) nextState = REQ;
                                end
                                REQ:
                                begin
                                        // vx assert("env1", clientRequest && !jobComplete);
                                        if (grant) nextState = GRANTED;
                                end
                                 VGRANTED:
                                begin
                                        // vx activate(dataOut);
                                        // vx assert("env2", !clientRequest);
                                        if (jobComplete || 'grant)
                                                nextState = NO_REQ;
                                end
                        endcase // case(state)
                end // else
        end // always

        always @(posedge clk)
        state <= nextState;
endmodule//BusInterface
```

FIG. 9B

```
module RoundRobinArbiter(clk, reset,
        dataA, requestA, grantA,
        dataB, requestB, grantB,
        dataC, requestC, grantC,
        writeData);

        input           clk, reset;
        input [0:0]     dataA, dataB, dataC;
        input           requestA, requestB, requestC;
        output          grantA, grantB, grantC;
        output [0:0]    writeData;

        // vx sentry dataA, dataB, dataC: clk;

        reg [0:0]       watchDogTimer, nextWatchDogTimer;
        reg [2:0]       state, nextState;
        // vx flop state, watchDogTimer;
        wire [0:0]      maxAccessTime;

        parameter       idle = 3'bOOO;
        parameter       ready = 3'b001;
        parameter       stateA = 3'b010;
        parameter       stateB = 3'b011;
        parameter       stateC = 3'b100;

        //next state logic
        always ©(reset or requestA or requestB or requestC or state or watchDogTimer)
        begin
                // vx deactivate(dataA,dataB,dataC);
                if (reset) begin
                          nextState= idle;
                          nextWatchDogTimer = 1 'b0;
        end
        else begin
                          next8tate= idle; // default state
                          //by default timer should increment
                          nextWatchDogTimer = watchDogTii-ner+1'bl;
```

FIG. 9C

```

                case (state)
                idle:
                begin
                        nextWatchDogTimer = 1'b0;
                        if (requestA || requestB || requestC) nextState = ready;
                end
                ready:
                begin
                        if (requestA) nextState = stateA;
                        else if (requestB) nextState = stateB;
                        else if (requestC) nextState = stateC;
                        else nextState = idle;
                end
                stateA:
                begin
                        // vx activate(dataA);
                        nextState= stateA;
                        // if request has been disabled or the max access time has
                        //reached change state.
                        if ((requestA == 1'b0) || (watchDogTimer == maxAccessTime)) begin

                                nextWatchDogTimer = 1 'b0;
                                if (requestB) nextState = stateB;
                                else if (requestC) nextState = stateC;
                                else nextState = idle;
                        end
                end
                stateB:
                begin
                        // vx activate(dataB);
                        nextState= stateB;
                        if ((requestB == 1'b0) || (watchDogTimer == maxAccessTime)) begin

                                nextWatchDogTimer = 1'b0;
                                if (requestC) nextState= stateC;
                                else if (requestA) nextState= stateA;
                                else nextState= idle;
                        end
                end
                stateC:
                begin
                        // vx activate(dataC);
                        nextState= stateC;
                        if ((requestC === 1'b0) || (watchDogTimer == maxAccessTime)) begin
                                nextWatchDogTimer = 1'b0;
                                if (requestA) nextState= stateA;
                                else if (requestB) nextState= stateB;
                                else nextState^ idle;
                        end
                end
            endcase//case(state)
        end
    end//always
```

FIG. 9D

```
          //state transition
          always @(posedge clk)
          begin
                    state <= nextState;
                    watchDogTimer <= nextWatchDogTimer;
          end
          //outputs
          assign grantA = (nextState == stateA);
          assign grantB = (nextState == stateB);
          assign grantC = (nextState == stateC);
          assign writeData = (grantA ? dataA : 'DATA_WIDTH'bz);
          assign writeData = (grantB ? dataB : 'DATA_WIDTH'bz);
          assign writeData = (grantC ? dataC : 'DATA_WIDTH'bz);
          assign maxAccessTime = •DEFAULT_MAX_ACCESS_TIME;
          // vx always onetrue("grant", grantA, grantB, grantC);
     endmodule // RoundRobinArbiter
```

FIG. 9E

FIG. 10

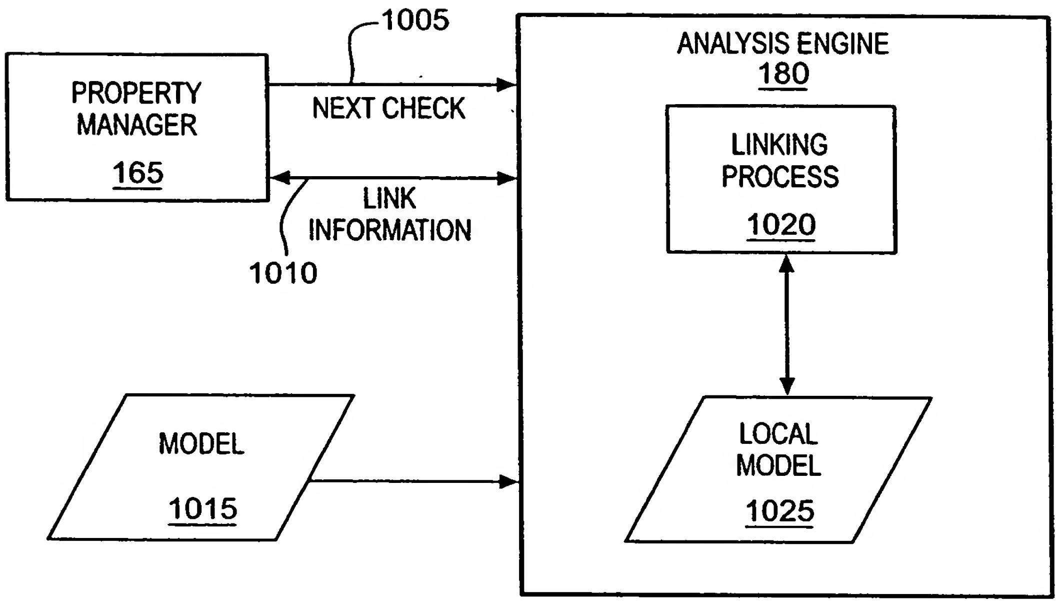

1005
PROPERTY MANAGER
165
NEXT CHECK
LINK INFORMATION
1010
ANALYSIS ENGINE
180
LINKING PROCESS
1020
MODEL
1015
LOCAL MODEL
1025

FIG. 11

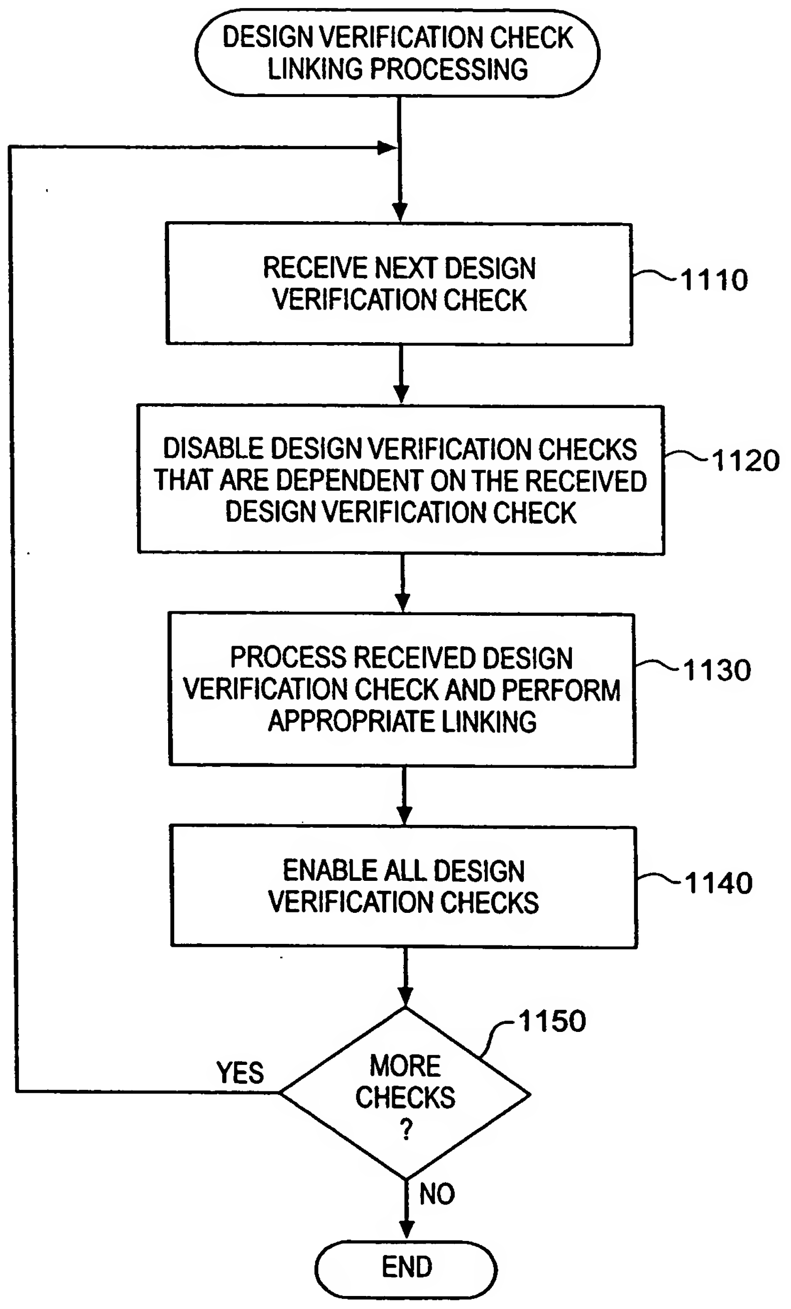
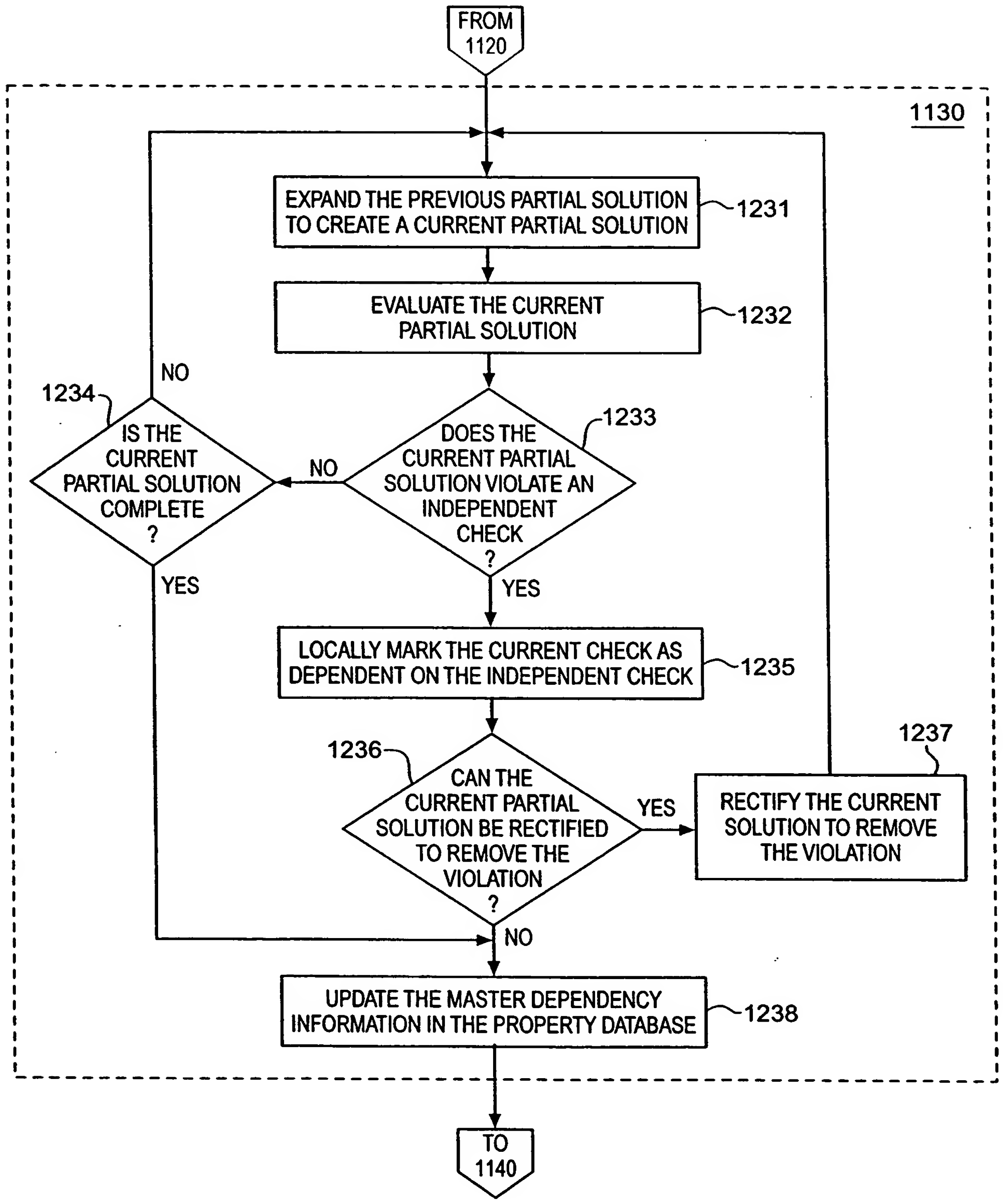

DESIGN VERIFICATION CHECK LINKING PROCESSING
RECEIVE NEXT DESIGN VERIFICATION CHECK
1110
DISABLE DESIGN VERIFICATION CHECKS THAT ARE DEPENDENT ON THE RECEIVED DESIGN VERIFICATION CHECK
1120
PROCESS RECEIVED DESIGN VERIFICATION CHECK AND PERFORM APPROPRIATE LINKING
1130
ENABLE ALL DESIGN VERIFICATION CHECKS
1140
MORE CHECKS ?
1150
YES
NO
END

FIG. 12


FROM 1120
1130
EXPAND THE PREVIOUS PARTIAL SOLUTION TO CREATE A CURRENT PARTIAL SOLUTION
1231
EVALUATE THE CURRENT PARTIAL SOLUTION
1232
NO
1234
IS THE CURRENT PARTIAL SOLUTION COMPLETE ?
1233
DOES THE CURRENT PARTIAL SOLUTION VIOLATE AN INDEPENDENT CHECK ?
NO
YES
YES
LOCALLY MARK THE CURRENT CHECK AS DEPENDENT ON THE INDEPENDENT CHECK
1235
1237
1236
CAN THE CURRENT PARTIAL SOLUTION BE RECTIFIED TO REMOVE THE VIOLATION ?
YES
RECTIFY THE CURRENT SOLUTION TO REMOVE THE VIOLATION
NO
UPDATE THE MASTER DEPENDENCY INFORMATION IN THE PROPERTY DATABASE
1238
TO 1140

```
===================================
Validation Results for Module "main"
===================================

========================
Functional Checks Summary
========================
Failed checks (FAILED)                                  = 5
Bounded failed checks (BOUNDED_FAIL)                    = 0
Inconclusive checks (INCONCLUSIVE)                      = 3
Secondary failed checks (SECONDARY)                     = 0
Interface checks (INTERFACE)                            = 12
Skipped checks (UNPROCESSED/DISABLED/DEFERRED)          = 0
Bounded passed checks (BOUNDED_PASS)                    = 0
Passed checks (PASSED/USER PASSED/CONDITIONAL)          = 144
___________________________________
Total checks                                            =164

==========================
RTL Characteristics Summary
==========================
Number of inferred flops             = 10
Number of inferred latches           = 0
Number of static X sources           = 0
Number of non-resettable flops       = 0

================================
Summary of failed checks by type:
================================
[CA] =0   (out of 39)
[BE] = 1   (out of 40)
[AX] = 1   (out of 15)
[CME]=0   (out of 10)
[CV] =0   (out of 29)
[AC] =0   (out of 7)
[AAO]=0   (out of 6)
[AID]=0   (out of 9)
[LVD] = 3   (out of 9)
```

FIG. 13A

```

Block enable [BE] : 1 failed check
------------------------------------
FAILED    BE  : arbiter.nextState_bit0_AX_6
              : Enable condition for block with assignment "arbiter.nextState_bit0_AX_6" is always off
              : Block with assignment "arbiter.nextState_bit0_AX_6" is defined on line 191 of "rr3.v"

Assignment execution [AX]  : 1 failed check
--------------------------------------------
FAILED    AX  : arbiter.nextWatehDogTimer_bit0_AX_1
              : Assignment "arbiter.nextWatchDogTimer_bit0_AX_1" only has a constant 1'b1 value
              : Assignment "arbiter. nextWatchDogTimer_bit0_AX_1" is defined on line 179 of "rr3.v"

Loss of valid data [LVD]  : 3 failed checks
--------------------------------------------
FAILED    LVD : dataA[0]
              : Loss of valid data has been detected on wire "dataA[0]"
              : Wire "dataA[0]" is defined on line 40 of "rr3.v"
              : VCD file is "roundRobin3/main/trace221 .vcd"
FAILED    LVD :dataB[0]
              : Loss of valid data has been detected on wire "dataB[0]"
              : Wire "dataB[0]" is defined on line 41 of "rr3.v"
              : VCD file is "roundRobin3/main/trace223.vcd"
FAILED    LVD :dataC[0]
              : Loss of valid data has been detected on wire "dataC[0]"
              : Wire "dataC[0]" is defined on line 42 of "rr3.v"
              : VCD file is "roundRobin3/main/trace225.vcd"
```

FIG. 13B

```
=========================================
Summary of bounded failed checks by type :
=========================================
[CA] =0  (out of 39)
[BE] =0  (out of 40)
[AX] =0  (out of 15)
[CME]=0  (out of 10)
[CV] =0  (out of 29)
[AC] =0  (out of 7)
[AAO] = 0  (out of 6)
[AID]=0  (out of 9)
[LVD] = 0  (out of 9)

=======================================
Summary of inconclusive checks by type :
=======================================
[CA] =0  (out of 39)
[BE] =0  (out of 40)
[AX] =0  (out of 15)
[CME]=0  (out of 10)
[CV] =0  (out of 29)
[AC] =0  (out of 7)
[AAO]=0  (out of 6)
[AID] = 0  (out of 9)
[LVD] =3  (out of 9)

Loss of valid data [LVD] 3 inconclusive checks
----------------------------------------------
INCONCLUSIVE  LVD    :busInterfaceA.dataOut[0]
                     : Valid data loss check on wire "busInterfaceA.dataOut[0]" did not complete
                     : Wire "busInterfaceA.dataOut[0]" is defined on line 80 of "rr3.v"
INCONCLUSIVE  LVD    :busInterfaceB.dataOut[0]
                     : Valid data loss check on wire "busInterfaceB.dataOut[0]" did not complete
                     : Wire "busInterfaceB.dataOut[0]" is defined on line 80 of "rr3.v"
INCONCLUSIVE  LVD    :busInterfaceC.dataOut[0]
                     : Valid data loss check on wire "busInterfaceC.dataOut[0]" did not complete
                     : Wire "busInterfaceC.dataOut[0]" is defined on line 80 of "rr3.v"

==========================================
Summary of secondary failed checks by type :
==========================================
[CA] =0  (out of 39)
[BE] =0  (out of 40)
[AX] =0  (out of 15)
[CME]=0  (out of 10)
[CV] =0  (out of 29)
[AC] =0  (out of 7)
[AAO]=0  (out of 6)
[AID]=0  (out of 9)
[LVD]=0  (out of 9)
```

FIG. 13C

```
116.  =====================================
117.  Summary of interface checks by type :
118.  =====================================
119.  [CA] =0  (out of 39)
120.  [BE] =0  (out of 40)
121.  [AX] =0  (out of 15)
122.  [CME]=0  (out of 10)
123.  [CV] =0  (out of 29)
124.  [AC] =6  (out of 7)
125.  [AAO]=0  (out of 6)
126.  [AID] =3  (out of 9)
127.  [LVD]=3  (out of 9)
128.
129.  Assertion correctness [AC] : 6 interface checks
130.  ----------------------------------------
131.  INTERFACE    AC    :busInterfaceA.envl
132.                     : Correctness check on assertion "busInterfaceA.envl" is at the interface
133.                     : Assertion "busInterfaceA.envl" is defined on line 121 of"rr3.v"
134.  INTERFACE    AC    :busInterfaceA.env2
135.                     : Correctness check on assertion "busInterfaceA.env2" is at the interface
136.                     : Assertion "busInterfaceA.env2" is defined on line 127 of "rr3.v"
137.                     : Check is used by conditional checks :
138.  AID                : arbiter.dataA[0]
139.  INTERFACE    AC    :busInterfaceB.envl
140.                     : Correctness check on assertion "busInterfaceB.envl" is at the interface
141.                     : Assertion "busInterfaceB.envl" is defined on line 121 of "rr3.v"
142.  INTERFACE    AC    :busInterfaceB.env2
143.                     : Correctness check on assertion "busInterfaceB.env2" is at the interface
144.                     : Assertion "busInterfaceB.env2" is defined on line 127 of "rr3.v"
145.                     : Check is used by conditional checks :
146.  AID                : arbiter.dataB[0]
147.  INTERFACE    AC    :busInterfaceC.envl
148.                     : Correctness check on assertion "busInterfaceC.envl" is at the interface
149.                     : Assertion "busInterfaceC.envl" is defined on line 121 of "rr3.v"
150.  INTERFACE    AC    :busInterfaceC.env2
151.                     : Correctness check on assertion "busInterfaceC.env2" is at the interface
152.                     : Assertion "busInterfaceC.env2" is defined on line 127 of "rr3.v"
153.                     : Check is used by conditional checks :
154.  AID                : arbiter.dataC[0]
155.
156.  Access of invalid data [AID] : 3 interface checks
157.  ----------------------------------------
158.  INTERFACE    AID   :dataA[0]
159.                     : Wire "dataA[0]" has accessed invalid data from the interface
160.                     : Wire "dataA[0]" is defined on line 40 of "rr3.v"
161.  INTERFACE    AID   :dataB[0]
162.                     : Wire "dataB[0]" has accessed invalid data from the interface
163.                     : Wire "dataB[0]" is defined on line 41 of "rr3.v"
164.  INTERFACE    AID   :dataC[0]
165.                     : Wire "dataC[0]" has accessed invalid data from the interface
166.                     : Wire "dataC[0]" is defined on line 42 of "rr3.v"
```

FIG. 13D

```

Loss of valid data [LVD] 3 interface checks
____________________________________________
INTERFACE    LVD        :arbiter.dataA[0]
                        : Wire "arbiter.dataA[0]" has loss of valid data at the interface
                        : Wire "arbiter.dataA[0]" is defined on line 141 of "rr3.v"
INTERFACE    LVD        :arbiter.dataB[0]
                        : Wire "arbiter.dataB[0]" has loss of valid data at the interface
                        : Wire "arbiter.dataB[0]" is defined on line 142 of "rr3.v"
INTERFACE    LVD        :arbiter.dataC[0]
                        : Wire "arbiter.dataC[0]" has loss of valid data at the interface
                        : Wire "arbiter.dataC[0]" is defined on line 143 of "rr3.v"

=========================================
Summary of unprocessed checks by type :
=========================================
[CA] =0  (out of 39)
[BE] =0  (out of 40)
[AX] =0  (out of 15)
[CME]=0  (out of 10)
[CV] =0  (out of 29)
[AC] =0  (out of 7)
[AAO]=0  (out of 6)
[AID]=0  (out of 9)
[LVD]=0  (out of 9)

[...]

===================================
Summary of passed checks by type :
===================================
[CA] =39  (out of 39)
[BE] =39  (out of 40)
[AX] = 14  (out of 15)
[CME]= 10  (out of 10)
[CV] =29  (out of 29)
[AC] = 1  (out of 7)
[AAO]=6  (out of 6)
[AID] =6  (out of 9)
[LVD] = 0  (out of 9)

[...]

=====================
List of inferred flops
=====================
busInterfaceA.state[0]
busInterfaceA.state[1]
busInterfaceB.state[0]
busInterfaceB.state[1]
busInterfaceC.state[0]
busInterfaceC.state[1]
arbiter.state[0]
arbiter.state[1]
arbiter.state[2]
arbiter.watchDogTimer[0]
```

FIG. 13E